April 24th, 2012

Spring 2012

University of Florida
EEL 4665/5666
Intelligent Machines Design Laboratory

Instructors:
Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

TA's:	Ryan Stevens
	Tim Martin
	Josh Weaver

[AUTONOMOUS RAIL INSPECTION]

GatorRail

Eitan Sisso
Student, University of Florida
Mechanical and Aerospace Engineering
esisso@ufl.edu

Table of Contents

3. Abstract

GatorRail is a mobile autonomous agent that scans railroad tracks for dangerous surface defects. The purpose of this project is to obtain proof of concept that defect detection can be managed using set of hacked dial indicators as linear displacement sensors. The sensor values from the indicators can be calibrated to a working value and then flagged every time it reaches below a critical value to signal a defect in the rail surface. The robot succeeded in detecting 100% of the surface defects after optimization and fine tuning. This project can be scaled and mass produced to facilitate low-cost, first pass railway surface analysis, especially in poorer, developing countries.

4. Executive Summary

GatorRail is a robot designed to inspect railway surfaces for any major imperfections and deficiencies. The autonomous device completes its mission by using a special array of modified dial indicators to detect these defects. The dial indicators consist of a tip that in constantly in contact with the rail surface, a spring that facilitates this constant contact, and a pin that presses on a membrane potentiometer as the tip displaces vertically; this essentially then acts as a detector of change in vertical displacement, and therefore the detection of a bump. Once bumps are detected a picture is taken, the distance from the starting point is recorded and so is the depth of the defect and this is all consolidated into one file.

The robot is made up of a lightweight, inflexible frame made out of t-slotted 80/20 aluminum extrusions that allow for quick adjustments. Additionally, a polycarbonate sheet is used to hold all the electronics on top of the robot. The structure of the robot is very small and compact and houses three main sensor systems as well as two motors. The sensor systems are the modified dial indicators described previously, an infrared sensor to detect the end of the railway segment and stop the robot before it runs off track, and hall effect quadrature encoders on the motors to determine position along the track.

Through experimentation error codes were considered and carefully placed to catch possible slip-ups during initialization of the device or during operation. This has prevented the robot from damaging itself or returning bogus data. During testing the motor speed was optimized to yield reliable results but also the fastest running speed possible; furthermore, calibration constant equations were resolved to yield defect depths in inches as opposed to just a voltage value. Overall, the robot performs well with 100% success rate at finding defects the last four test runs.

5. Introduction

Railroads are one of the oldest and most reliable forms of transportation in the modern era; however, railroad tracks can also be unsafe if not maintained properly. The metal on metal contact from train wheels with the tracks and the weight of the trains can lead to the development of surface faults which can ultimately lead to fracture and catastrophic train accidents as depicted in Figure 1. The main purpose of this project is to provide a means of monitoring railways autonomously for these defects.

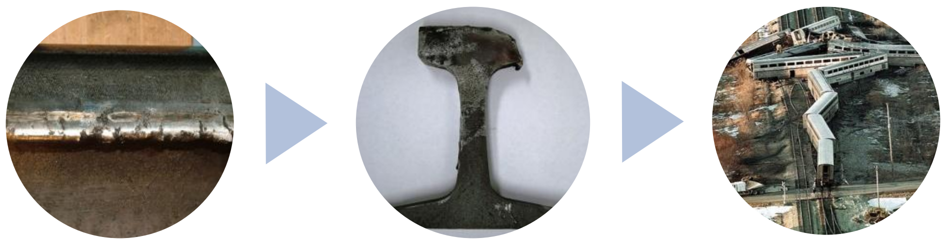


Figure 1: Progression of surface cracks that can lead to catastrophic events

The device, GatorRail, named that way since it is being developed at the University of Florida, will be in charge of monitoring railroad surface conditions. The task will be accomplished by running along the train tracks with several finger-like dial indicators that will run along the surface detecting any sharp vertical displacement. Because of the noise in recorded data, the robot will not be able to differentiate small surface defects from vibrations; however, these smaller imperfections will be considered negligible and will only be of concern once they've grown to a larger, more damaging size. In order to perform this task reliably, it must meet some special requirements as outlined:

GatorRail shall be able to...

- … detect surface defects on railways
- … have a very rigid frame to reduce vibrations and therefore noise in data
- … fit snuggly on a standard railroad track so that it will not fall or veer off course
- … protect the tip of the dial indicator from snagging because the sensor might not be reliable if the tip is damaged
- … be small and easily carried by a single technician
- … be cost affordable and easy to reproduce

… relay back important defect information (position and depth) to the main computer
… recognize when it reaches the end of a test rail to prevent damage

GatorRail should be able to...

… take a picture of the defect and relay it back to the main computer
… take into account the resonance frequency of the frame and the motors into the sensor
… optimize its speed and the speed of data acquisition in order to ensure no sacrifices are made in performance or reliability
… be adaptable to different track widths in case it is to be used internationally

Design Specifications

6. Integrated System

GatorRail is an autonomous (self-driven) device; therefore, it must have a processing unit to be able to sense and react accordingly to different stimuli in the environment and to incorporate all of the different subsystems into one cohesive device. GatorRail will use the Epiphany DIY microprocessor board developed by Tim Martin & Out of the Box : Electronics and Robotics LLC to manage all the data and consolidating all the subsystems.

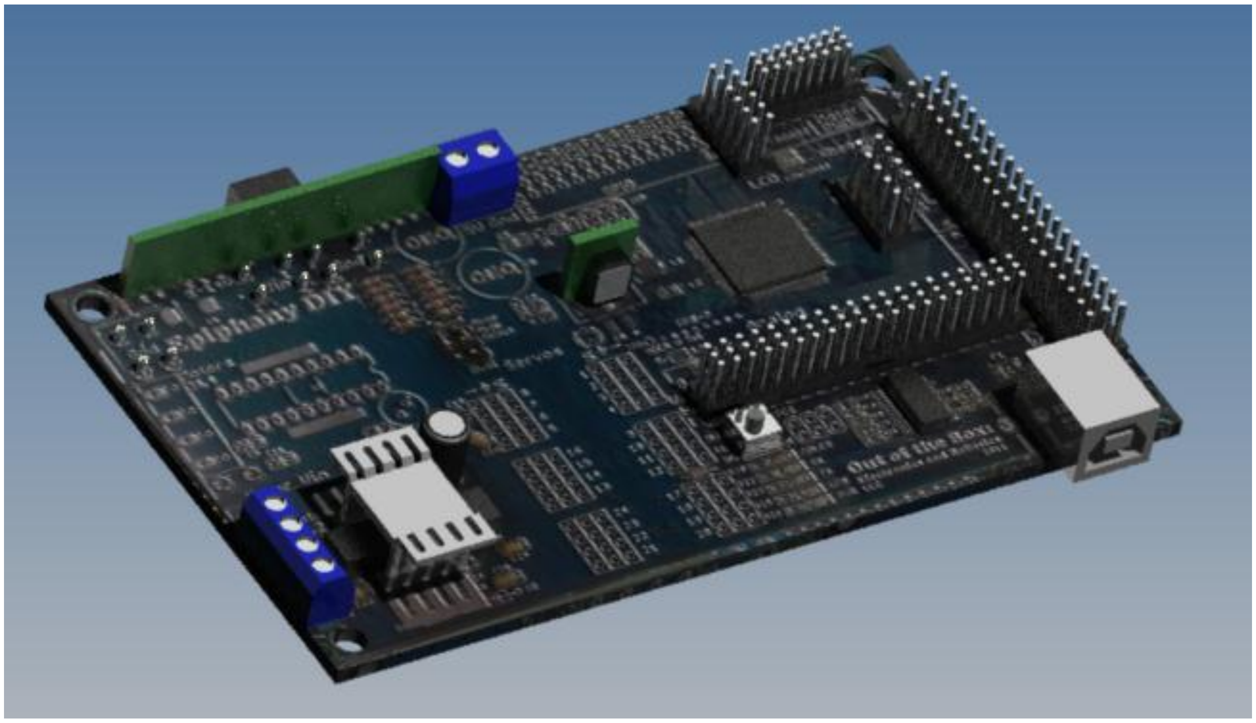


Figure 2: CAD Model of the Epiphany DIY Board

The Epiphany DIY board sports an 8-bit AVR XMega A1 microcontroller (ATxmega64A1) Some of the key features of the Atxmega64A1 are:

Flash (Kbytes): 64 Kbytes

Pin Count:100

Max. Operating Frequency: 32 MHz

CPU:8-bit AVR

Max I/O Pins: 78

All of the data acquisition, except for the encoders, will be coming in from the ADC Channels built into the Epiphany DIY Board. The encoders will be using I/O Pins to capture pin changes and generate events and interrupts.

Using this microcontroller and board setup, GatorRail will perform the rail monitoring task. Shown below in Figure 3 is the flowchart of behaviors of the robot.

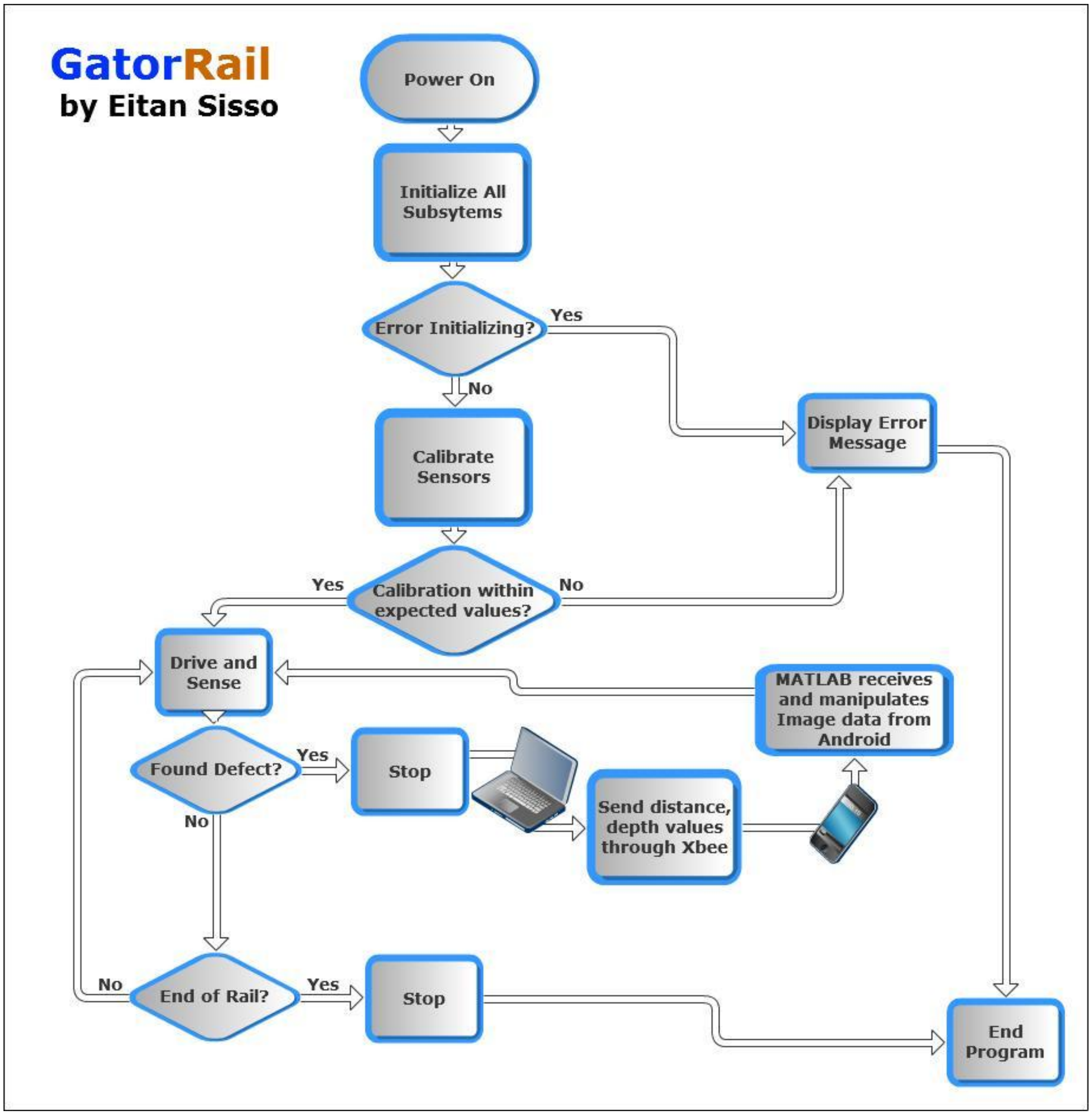


Figure 3: GatorRail Behavior Flowchart

The other important part in the robot crucial to system integration is the LCD screen. This allows for debugging of the robot and display of error messages in case anything goes wrong.

The LCD used for this robot is a Basic 16x2 Character LCD - Amber on Black 3.3V from Sparkfun.com. The LCD pins interfaced directly with the LCD pins on the Epiphany DIY board so no modifications were required, only minor soldering.

Figure 4: 16x2 LCD Screen

The LCD pinout is presented in the Appendix section.

All the GatorRail subsystems (chassis, sensors, actuators, LCD and microprocessor board) which will be covered in detail were then assembled and came together into one main platform as shown in the CAD rendering and actual photos of the robot in Figure 5.

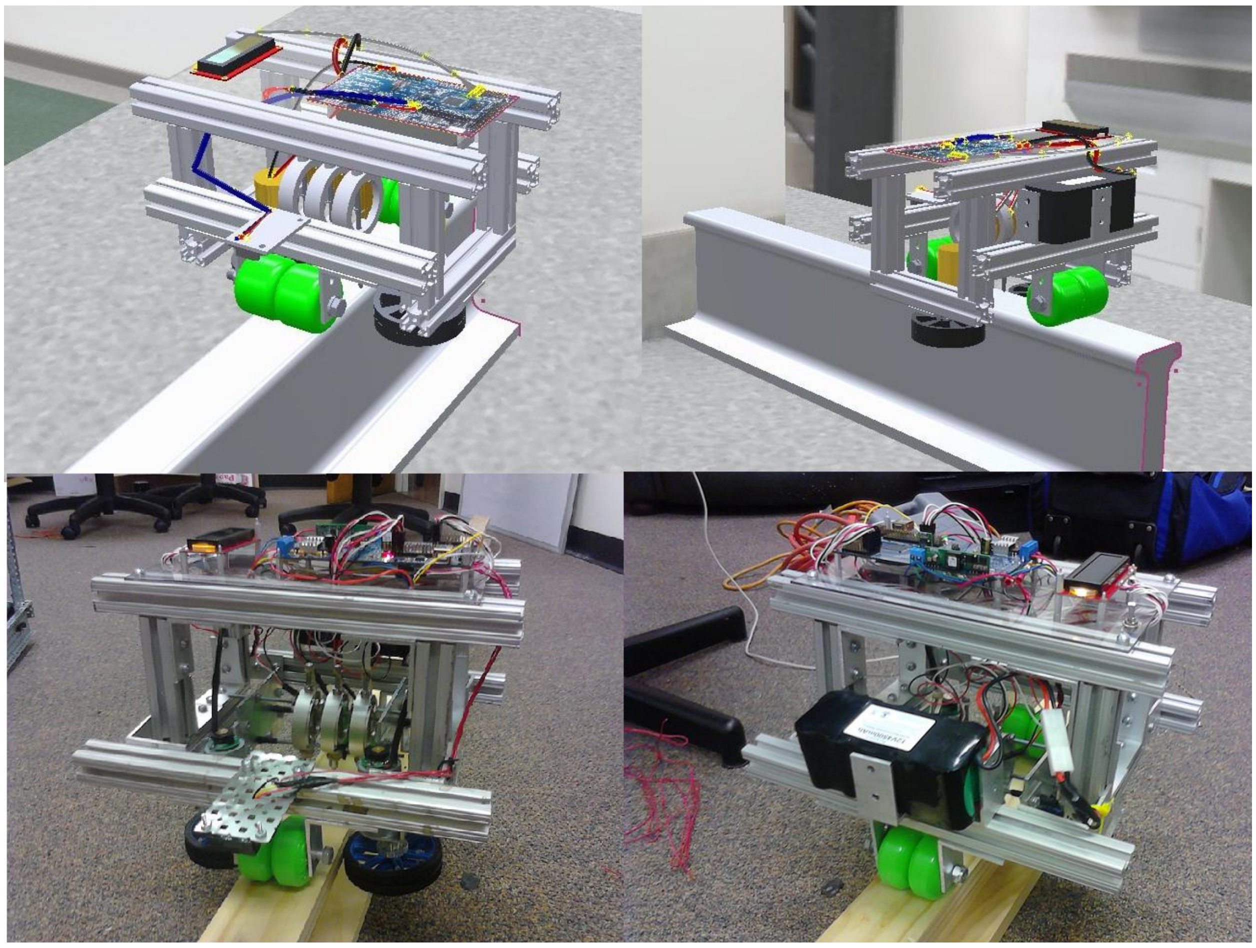


Figure 5: CAD Model comparison to completed model

7. Mobile Platform

a. Chassis

The chassis is composed mainly of 1”x1” 80/20 Aluminum T-slotted extrusions, 2”x2”1/8” L-Brackets, and a 5.5”x10” sheet of Polycarbonate Lexan®. The chassis was designed in such a way that allows for many quick adjustments. Due to the nature of the T-slotted 80/20 frame, L-brackets can slide back and forth speedily when loosened. Furthermore, the L-brackets were all standardized and almost every component on the robot (even the battery holder and wheel supports) use the L-brackets to carry out their function; this simplified the manufacturing process and would allow for rapid reproduction of the design. Lastly, the polycarbonate sheet was chosen to isolate the main electronics from the aluminum frame and it also gives a sneak peek into the insides of the robot.

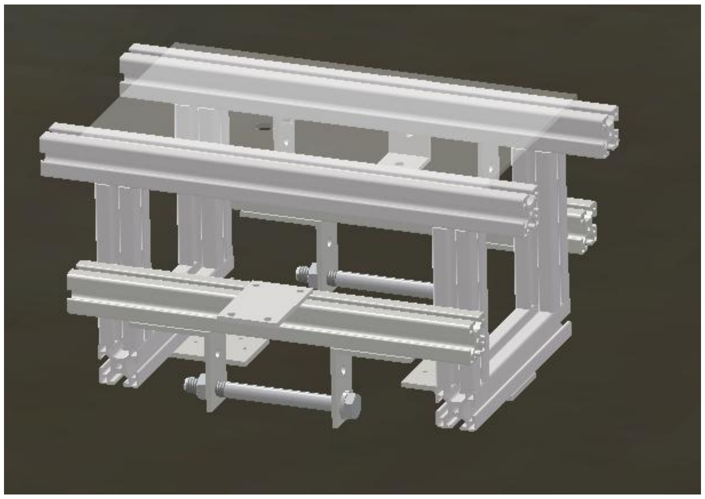


Figure 6: GatorRail Chassis

b. Drive Configuration

The drive train consists of two main drive motors on either side of the track propelling the device forward. The wheels on top of the surface will rotate freely and provide stability. This design will help the robot be very rigid and stable; furthermore, it will keep it on the track whenever banks or curves are reached.

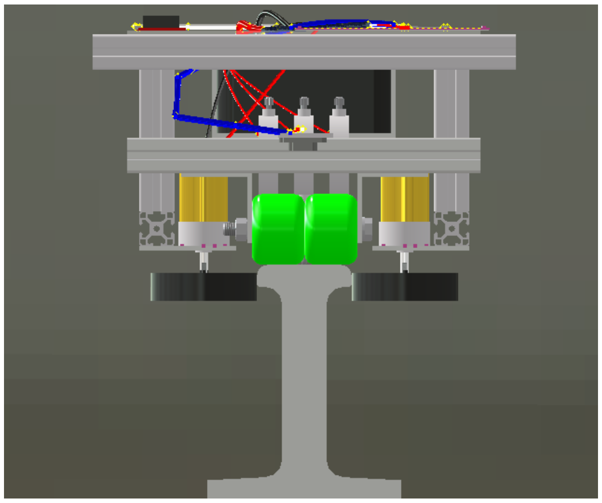


Figure 7: Wheel Configuration

The actuated wheels used to drive forward will be two 80mm x 10mm (black in Figure 7) wheels stacked together and the freely rotating wheels will be two 50mm x 35mm (green in Figure 7) wheels next to each other.

More information on the motors used for motion up and down the rail will be provided in the actuation section of the report.

8. Actuation

a. Drive Motors

Objective: To be able to move forward and backwards on the tracks at a controllable rate

Actuation Type: Two DC Gear Motors

Motor Specs:

Manufacturer: Pololu

Gear Ratio: 131:1

Free run speed @12V: 80 rpm

Free run current @12V: 300 mA

Torque @12V: 250 oz-in

Additional Features: Built in 64 CPR Hall-Effect Quadrature Encoder

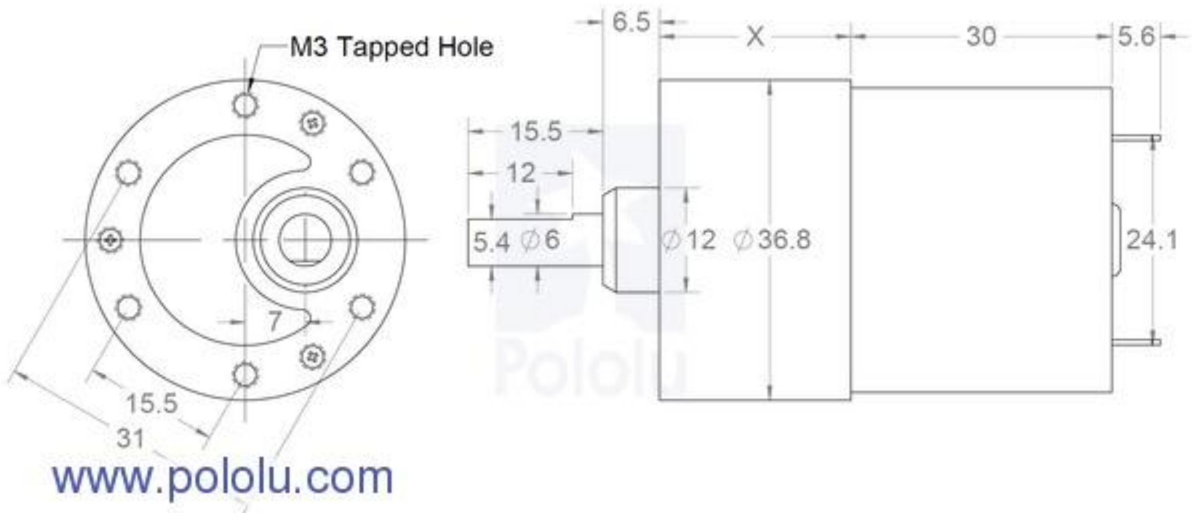


Figure 8: Drive Motor Drawings

The reasoning behind this choice was that the project called for a motor that was not too fast but also had a considerable amount of torque to propel the body of the robot forward along the track. Before purchasing these motors, I had originally purchased Pololu High Power Micro Metal 298:1 Gearmotors, these had a stall torque of 70 oz-in as opposed to the final design's 250 oz-in.

An important lesson came about from this, and that is to estimate the weight and friction of the robot before purchasing motors.

A wiring table for the motor is presented below:

Table 1: Motor Wiring Table

		Motor 1 (Right Drive Motor)	**Motor 2 (Left Drive Motor)**
Color	**Function**	**Connect to:**	**Connect To:**
Black	Motor Power	Epiphany DIY M1 -	Epiphany DIY M2 +
Red	Motor Power	Epiphany DIY M1 +	Epiphany DIY M2 -
Blue	Hall Effect sensor Vcc	Epiphany DIY +5V Regulator	Epiphany DIY +5V Regulator
Green	Hall Effect sensor GND	Epiphany DIY GND	Epiphany DIY GND
Yellow	Encoder Output A	Epiphany DIY Port D Pin 0	Epiphany DIY Port D Pin 0
White	Encoder Output B	Epiphany DIY Port D Pin 0	Epiphany DIY Port D Pin 0

9. Sensors

a. Fault Detector (Modified Dial Indicator)

The dial indicator sensor will be hacked to give the microprocessor data in order to detect cracks. The indicator consists of a spring, guide pin, guide rail, housing, and the indicator tip. A membrane potentiometer will be added inside of the guide rail, to be actuated by the guide pin, to measure the deflection of the indicator tip (as seen on Figure 4).

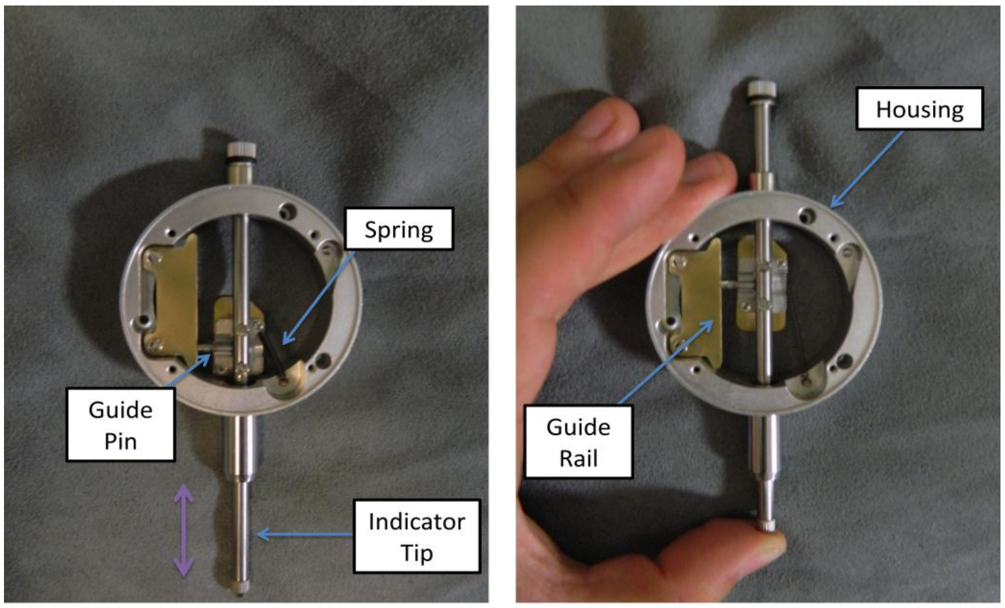


Figure 9: Digital Indicator Taken Apart

1. Membrane Potentiometer

A membrane potentiometer is a potentiometer that works without the need of a rotating wiper. The wiper assembly is instead replaced with anything that can push down with enough force to connect the collector membrane to the resistor membrane. A layer by layer description is shown in Figure 10.

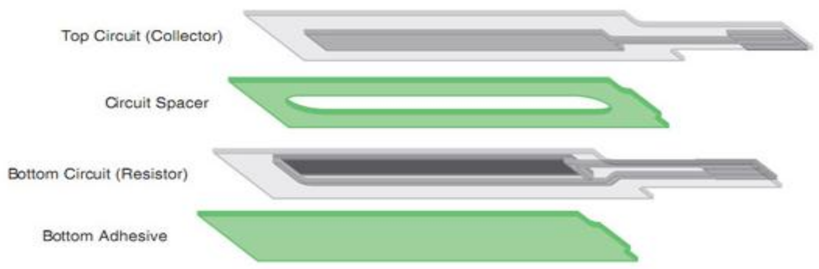


Figure 10: Membrane Potentiometer (Layers)

2. Spring

A test was done on the spring of the dial indicator using an Instron (Figure 7) to determine the spring constant. The spring constant was determined to be approximately 0.057 N/mm as shown in Figure 6. Unfortunately, the membrane potentiometer requires 1-3 N of pressure to be applied in order to yield a useful value instead of garbage noise and the spring did not have enough force to apply that lateral pressure and still allow the indicator tip to freely slide back and forth. Sequentially, a rubber band was added to the top of the dial indicator; therefore, providing a constant down force that aids the spring in its function.

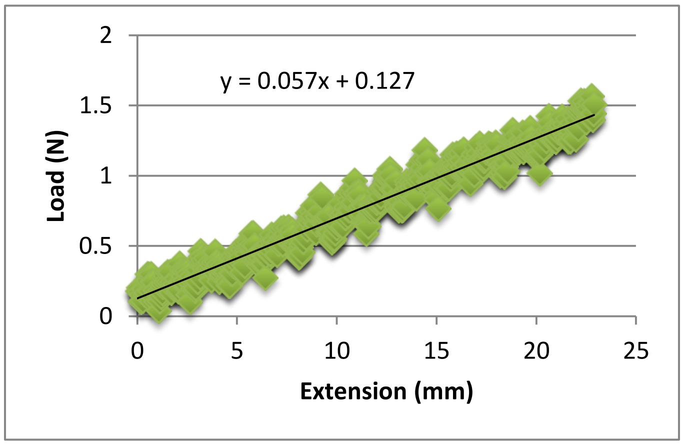


Figure 12: Spring Test Results

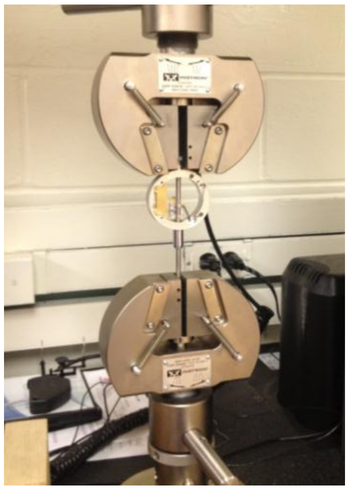
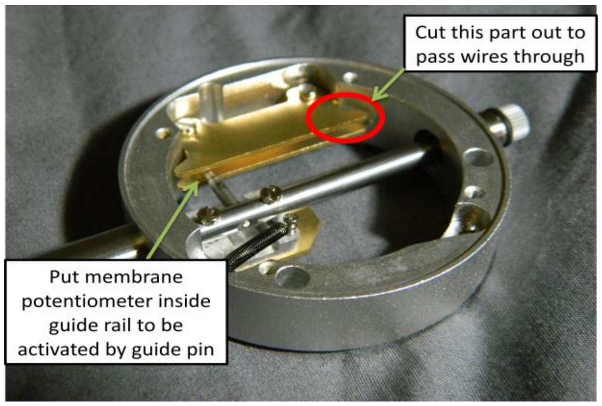


Figure 11: Instron Tensile Testing Machine

3. Guide Rail & Guide Pin

The guide rail and guide pin will house the hack for the mechanical system, the membrane potentiometer, depicted in Figure 13.


Cut this part out to pass wires through
Put membrane potentiometer inside guide rail to be activated by guide pin

The guide pin will actuate the membrane potentiometer by exerting the necessary 1-3 Newtons of force to have the collector come in contact with the resistor. This relationship can be seen in Figure 14.

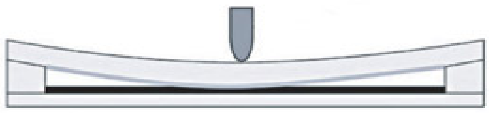


Figure 14: Guide Rail Modifications

Figure 13: Guide Pin Actuating Membrane Potentiometer

The modified dial indicator is publicized below in Figure 10 along with a graph that shows voltage vs. time during which the indicator tip was moved up and down. Note in the voltage graph the thick blue noisy data happened when the guide pin was not exerting the minimum 1-3 Newtons of force on the membrane potentiometer.

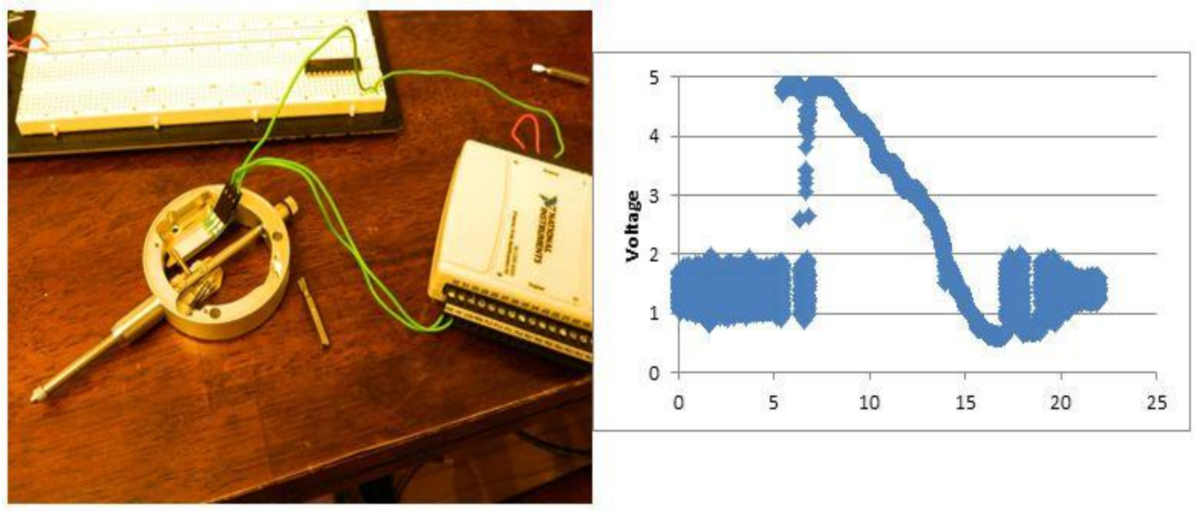


Figure 15: Modified Dial Indicator with DAQ to test output signal Voltage

4. Housing

The housing will play the critical role of affixing the whole system to the robot and also keeping all the sensor pieces in place. In order to connect the sensors to the chassis rigidly, existing holes previously used to mount the face of the indicator will be bored out to a larger size and threaded to pass a threaded rod through. Three of such connections will be made for every indicator; hence, they will be very stiff when running along the track. Two out of the three completed connections are shown in Figure 11 below.

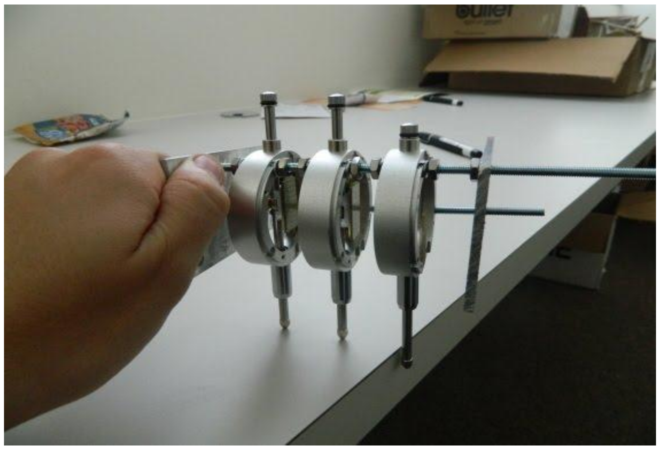


Figure 16: Housing with two of three threaded connections installed

5. Indicator Tip

The indicator tip has a smooth rolling surface in the bottom that will allow it to glide along the railroad track surface seamlessly. Furthermore, the rounded tip geometry will prevent snagging in the defects.

6. Wiring Schematic

Each modified dial indicator will have 3 pins going out. The indicators will be wired to the Analog to Digital Port A in positions 3,4,5 corresponding to left, middle, and right respectively. The GND pin should be connected to the (-) pin in the Epiphany DIY board, the Collector pin should be connected to the (S) pin in the DIY board, and the V+ pin should be connected to the (+) pin of the DIY board

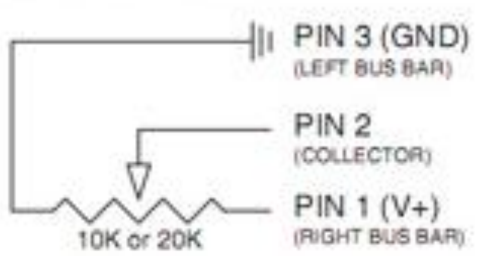


Figure 17: Wiring Schematic for the Membrane Potentiometers

b. End of Rail Detection (Infrared Sensor)

The end of rail detection will be the "obstacle avoidance" of the robot. Because of the circumstances that GatorRail will not have an infinite rail road track to be tested and demoed on, the robot will use an infrared sensor to determine whether there is a railroad track ahead or not.

Sensor Specs

Model: Infrared Proximity Sensor - Sharp GP2Y0A21YK

Range: 6cm-80cm

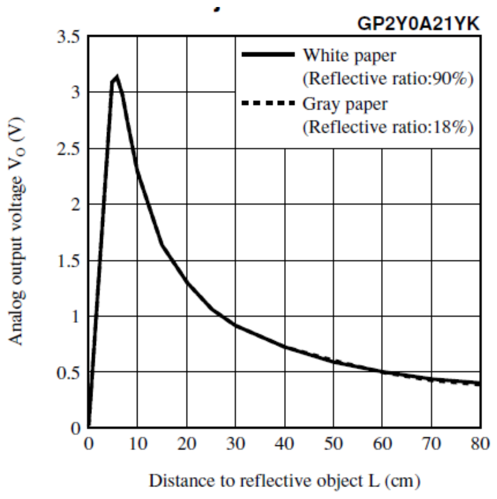


Figure 18: Infrared Voltage vs. Distance Graph

The infrared sensor will be placed on a platform ahead of the robot which will face directly down as shown in Figure 19; whenever the reading changes to show that there is no longer a surface ahead (the rail segment has ended, then the robot will immediately stop and end the rail analysis)

The sensor connects directly to the Analog to Digital Converter Port A pin 7

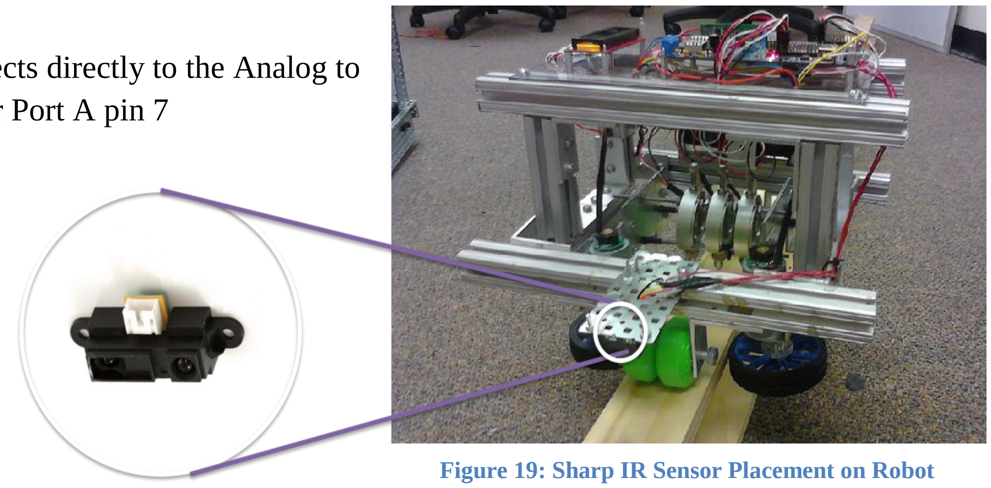


Figure 19: Sharp IR Sensor Placement on Robot

c. Encoder

The encoder will provide a position along the railroad track. This will be useful for tracking at what position the defect is on the track. It will also allow for proper movement when aligning the camera shot.

Encoder Specs

Model: 64 CPR Quadrature Encoder

Count Per Revolution (no gearing): 64

Count Per Revolution (with 131:1 gearing): 8400

Method: Two channel Hall Effect

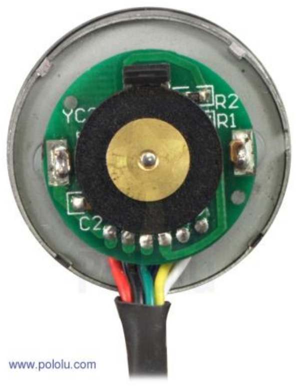
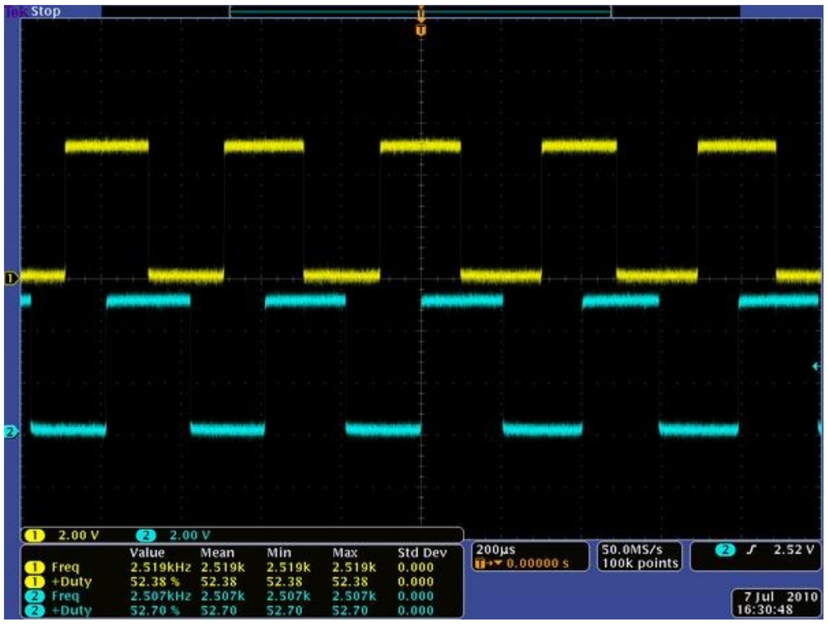


Figure 20: 64 CPR Encoder Mounted on Motor[l]

The sensor output comes from two channels that are off phase by 90 degrees and the angle can be calculated from the count of every rising and falling edge. Also, by seeing if Channel A or Channel B rise first, one can also determine the direction of travel from the encoder. Below is a sample of the data obtained from the encoder using an oscilloscope.



```
            Value     Mean    Min     Max     Std Dev
1 Freq      2.519kHz  2.519k  2.519k  2.519k  0.000
1 +Duty     52.38 %   52.38   52.38   52.38   0.000
2 Freq      2.507kHz  2.507k  2.507k  2.507k  0.000
2 +Duty     52.70 %   52.70   52.70   52.70   0.000
```

Figure 21: Encoder Data

The encoder wiring schematic is the same as the motor's shown in Table 1 previously.

d. Android Camera

GatorRail will use an android phone mounted in the back as an image capture device to take pictures of the railroad track defects when they occur. It will be communicating through an internet protocol webcam application where the computer will take a snapshot when a defect is found. A sample image can be seen in Figure 23.

Camera Phone Specs:

Model: Huawei Ascend M860

Operating System: Google Android 2.2 (API 8)

Camera Resolution: 3.2 megapixels

Processor: 528 MHz Qualcomm S1 MSM7625

Application used: IP Webcam by Pas

Figure 22: Huawei Ascend M860

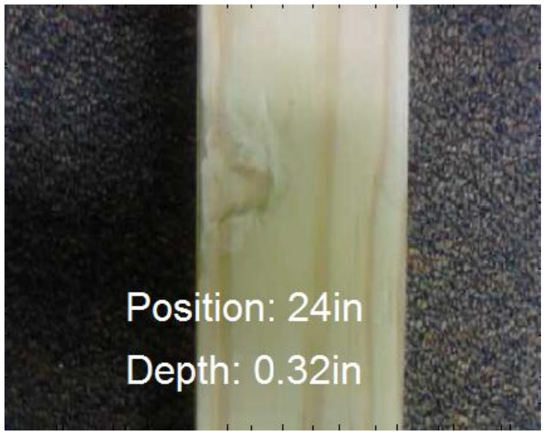


Figure 23: Sample image of Android capture and position and depth superimposed onto image using MATLAB

9. Communication Modules

a. Xbee (Robot to PC)

An Xbee communication system was established to transmit data from the robot to the PC. The Xbee used was the Parallax 32440-RT which has a built in antenna. The PC Connector has a USB dongle which allows for the creation of a virtual serial port.

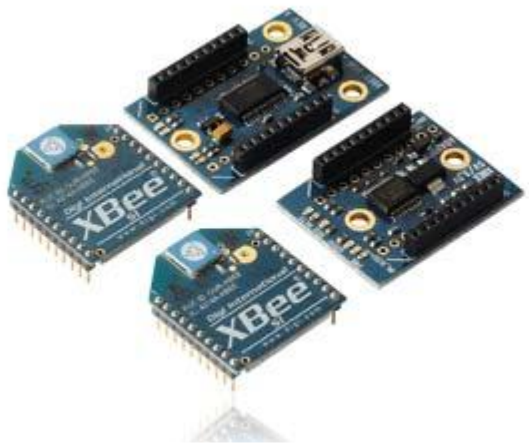


Figure 24: Xbee Module for PC and for Epiphany DIY Board

Xbee Module Specs

Model #: Parallax 32440-RT

Type of Communication: Virtual Serial

Range: 400 ft Line-of-sight

Baud Rate: 9600

Data Bits: 8

Stop Bits: 1

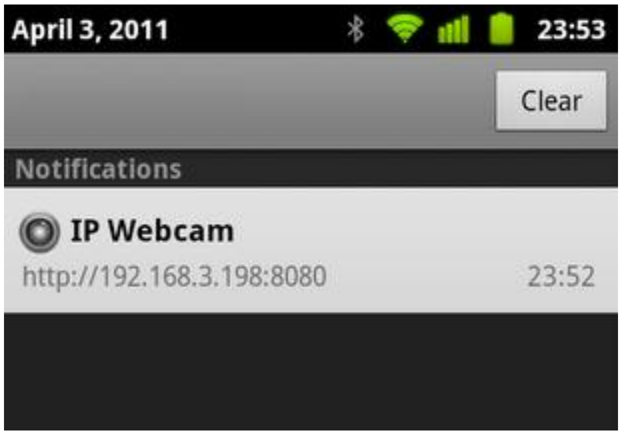
Terminator: Carriage Return

COM Port Used: 3

The Xbee signals are received into the PC through MATLAB and processed according to the Report behavior as specified in the next section of the report.

b. Wi-fi (Android Smartphone to PC)

The Android camera phone is used to snap a picture using the phone's camera and transmit the data through an IP (internet protocol) address. To the right is a sample IP address that could be used to retrieve the camera stream. Because the stream is simply a set of jpegs compiled together to create a video, when MATLAB receives the signal that a defect was found, MATLAB pulls the next frame from the stream



IP Webcam

http://192.168.3.198:8080 23:52

Figure 25: Sample IP Address from the IP Webcam App

10. Behaviors

a. Calibration and Initiation

The calibration and initiation behavior initiates all necessary systems of the robot like the clocks, counters, interrupts, LCD module, USB module, and Xbee Module. Furthermore, it takes an average value of the sensor readings to determine the calibrated value. If any of the setups fail or the calibration value are not close to what is expected, GatorRail returns the appropriate error message through the LCD to allow the user to quickly debug. Once it is done getting ready, it goes right into the Driving and Data Collection Behavior

b. Driving and Data Collecting

GatorRail will be programmed to drive down the rail and collect data from all sensors as it drives. The data will be coming in from the two main sensor systems: the dial indicators and the infrared sensor rangefinder. The behavior will be to drive forward until a surface defect is detected or the end of rail is found. At the same time as the robot drives forward there will be an event driven interrupt system that will keep track of wheel revolutions and relative wheel angle data coming in from the encoder.

c. Surface Flaw Discovery

This behavior is triggered when surface faults are detected. This happens when the sensor value of one of the dial indicators drops more than 10% of the calibrated value. If a crack is detected, the robot will come to a complete stop. The distance marker will be sent through Xbee to MATLAB in the main computer. The data will also include and indicator to tell the camera to snap a picture and the tip deflection/ the depth of the fault. The depth data will be used to determine if the crack needs immediate attention or it can be monitored through the years. After all of these tasks are completed the robot will return to behavior (b) Driving and Data Collecting

c. End of Rail Detection

If the infrared sensor value drops significantly this means that the rail is no longer under the robot and it must stop immediately to prevent from running off the test track. Once this behavior is triggered, a command is sent through MATLAB through Xbee to quit the MATLAB code and let the computer know the end of rail has been reached; also, a message is displayed on the LCD "GatorRail Analysis Complete"

11. Experimental Layout and Results

The experimental procedures in this project constituted of testing the software and coding to ensure proper functionality and also to optimize the performance of the robot.

a. Conversion factor for the depth measurements

The conversion factor for the depth measurement was calculated by deflecting the dial indicator tip known amounts measured with a caliper and writing down the corresponding decimal value output from the analog to digital converter channel. The results were then recorded into Excel and a conversion factor was determined from the slope of the graph:

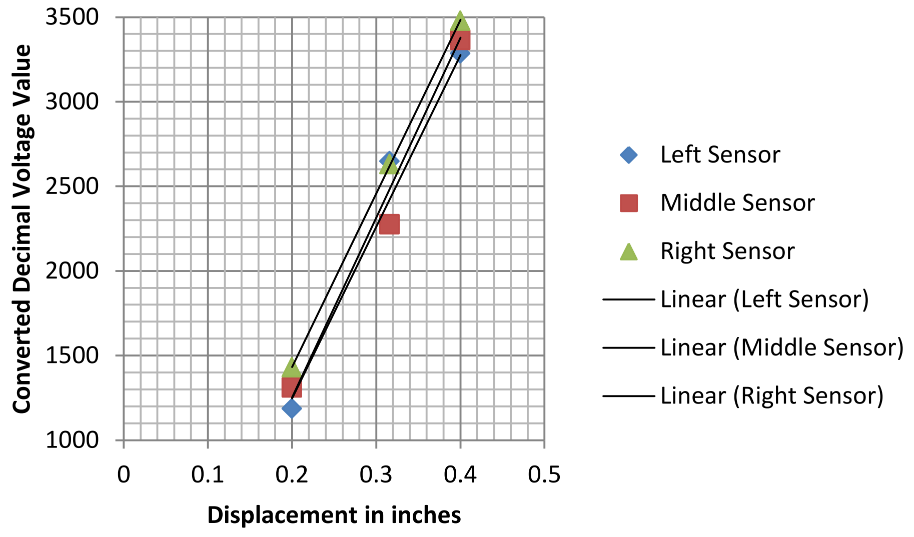


Figure 26: Experimental Results for Depth Measurement

The conversion equations are as follows:

Left Sensor Depth (in.) = (decimal value of converter voltage - 873.06) / 10629

Middle Sensor Depth (in.) = (decimal value of converter voltage - 783.07) / 10150

Right Sensor Depth (in.) = (decimal value of converter voltage - 617.67) / 10252

b. Optimizing the motor speed for best results

The motor speed was optimized by calculating the percentage success rate of detecting a defect at different motor duty cycles. Every time the robot was reset at different duty cycles and allowed to have four runs. If a defect was not found or skipped, the robot dropped in percentage rate. In the test there were 3 defects on the rail. The results are shown in Table 2 and Figure 27.

Table 2: Motor Duty vs. Percentage Success Rate

Motor Duty	1st Run	2nd Run	3rd Run	4th Run	Average
1024 (100%)	66.66667	66.66667	100	66.66667	75
900 (87.8%)	100	66.66667	100	100	91.66667
800 (78.1%)	100	100	100	100	100
700 (68.4%)	100	100	100	100	100

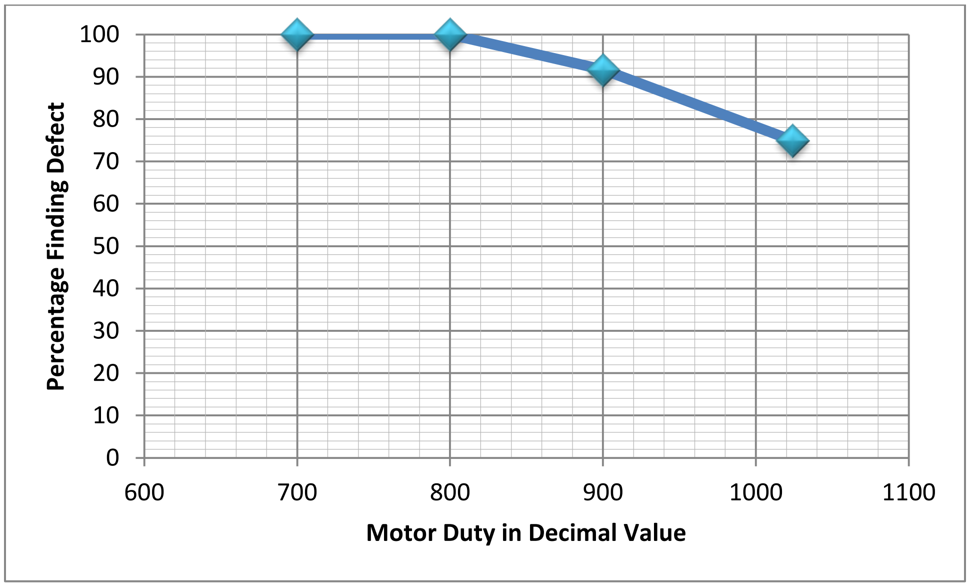


Figure 27: Average Success Rate vs. Motor Duty Cycle

The results indicate that a 78% motor duty cycle will yield reliable results and is the fastest the robot can run at to achieve reliable detection of surface rail flaws.

CLOSING

12. Conclusion

In final analysis, GatorRail has been able to successfully meet all of the shall requirements as defined in the Introduction section of this report. GatorRail successfully detects surface defects at an optimized speed, it has a rigid and lightweight frame, it can relay important information back to the main computer through wireless Xbee communications, and it can detect when it has reached the end of the rail and stops before running off the track. Additionally, it can also fulfill some of the should requirements like taking a picture with the Android phone and relaying that back to the computer, and having an adaptable frame that can be quickly modified to fit different track sizes.

Despite the success of the project, it did have some limitations. The main drawback was money. Unfortunately, funds are not infinite and the components chosen were not always the best performing ones but the ones that fit within the allotted budget. Another limitation was resources. A lot of the parts for the robot could have been machined using a CNC machine and would have yielded impeccable fits; however, the robot was manufactured using bandsaws, drill presses, and handheld tools which have a much higher dimension tolerance and result in less accurate parts. Also, programming knowledge was a big limitation personally. Because of my lack of previous programming experience some of the code could have been more complex; despite running well, the code could have been optimized to use less clock cycles if I knew more.

GatorRail performed quite well in all areas of operation, there are however a few things that stood out in a positive sense and a few things that did not necessarily meet expectations. The biggest pleasant surprise was the amazing resolution that the robot can get from the modified dial indicators. In just a short 0.4 in of displacement the value changed from 5V all the way down to 0.75V, this allowed for much more accurate depth measurements than initially anticipated. On the other hand, being able to only have 3 modified dial indicators due to space issues (only so many can fit on a track) was disappointing. I would have liked to place at least 5 indicators to cover a larger area of the track; however, the three in place do a great job at detecting faults.

Looking towards the future of this project, I'd like to replace the Xbee module to an Xbee to Bluetooth module to communicate directly with the Android device and have the Android device relay information back through a text message or email. Another neat addition would be to mark the rails with distance markers and run a detection code using the camera, this would allow for absolute positioning instead of only relative positioning. Also, I'd like to make the frame a little less adjustable because having it be so adjustable created some issues along the build. Lastly, I would have liked to have started the quadrature decoder programming much earlier and I recommend that anyone that tries to do quadrature decoding should probably start early because it is much more complicated than it seems.

13. Documentation

[1] J. Esposito. (2009). *Serial Communication in Matlab V2* [Online]. Available: http://www.scribd.com/doc/44954748/44316459-Serial-Communication-in-Matlab-V2

[2] T. Martin. (2012, April). *Epiphany DIY* [Online]. Available: http://sites.google.com/site/epiphanydiy/home

[3] Atmel Corporation. (2012, April). *ATxmega64A1- Atmel Corporation* [Online]. Available: http://www.atmel.com/devices/ATXMEGA64A1.aspx?tab=documents

[4] Sparkfun Electronics (2012, April). Basic 16x2 Character LCD - Amber on Black 3.3V - SparkFun Electronics: [Online]. Available: http://www.sparkfun.com/products/9054

[5] Spectra Symbol. (2012, April). *Thinpot - Spectra Symbol* [Online]. Available: http://www.spectrasymbol.com/potentiometer/thinpot

[6] Sparkfun Electronics. (2012, April). *Infrared Proximity Sensor - Sharp GP2Y0A21YK* [Online]. Available: http://www.sparkfun.com/products/242

[7] Pololu. (2012, April). *131:1 Metal Gearmotor 37Dx57L mm with 64 CPR Encoder* [Online]. Available: http://www.pololu.com/catalog/product/1447

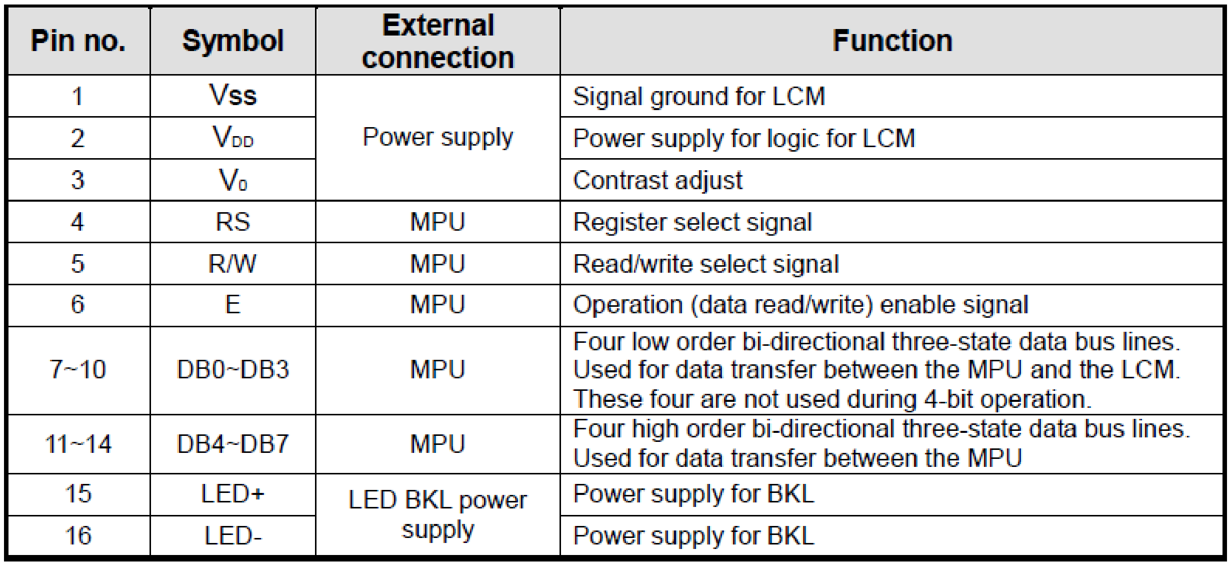
[8] Google Play. (2012, April). *IP Webcam - Android Apps* [Online]. Available: https://play.google.com/store/apps/details?id=com.pas.webcam&hl=en

[9] Parallax Inc. (2012, April). *XBee 1mW PCB Antenna* [Online]. Available: http://www.parallax.com/Store/Accessories/CommunicationRF/tabid/161/ProductID/638/List/0/Default.aspx?SortField=ProductName,ProductName

14. Appendices

a. LCD Pinout

16x2 Character LCD - Amber on Black 3.3V



Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL

b. MATLAB Code

```
%%The Following Code will receive Serial data and manipulate the image from
%%the android

error_count=1;
tline= nan;
notdone=1;

port = strcat('COM',num2str(3) );

out = instrfind('Port', port);  % Check to see if THAT serial port is
                                % already defined in MATLAB
if (~isempty(out))  % If it is
    disp('WARNING:  port in use.  Closing.')
    if (~strcmp(get(out(1), 'Status'),'open'))  % Is it open?
        delete(out(1)); % If not, delete
    else  % is open
        fclose(out(1));
        delete(out(1));
    end
end

%%Initialize Serial Communication
serialport = serial('com3');
set(serialport,'baudrate', 9600)
set(serialport, 'databits',8);
set(serialport,'stopbits', 1);
set(serialport, 'flowcontrol', 'none');
set(serialport, 'terminator', 'CR');
set(serialport, 'timeout',.5);
fopen(serialport)
pause(1)  % pause for a second to start getting data
disp('Xbee Initialized')

 while (notdone)
  pause(0.2)
%% IF THERE IS NO DATA this loop will not run
  while(~serialport.BytesAvailable==0)
    % read until terminator
    tline = fscanf(serialport,'%s');
    pause(0.1)
    [new,encoder_val,depth] = strread(tline,'%s%f%f', 1, 'delimiter', ',');
    tline= cell2mat(new);
    if(tline=='Found')
        %% This part takes in IP webcam data
        url = 'http://10.128.62.111:8080/shot.jpg'; %% Change this code to
reflect actual IP Address of Camera
        im  = imread(url);
        image(im);
        % Text at arbitrary position
        text('units','pixels','position',[100
100],'fontsize',24,'color','w','string','string',[ 'Position: '
num2str(encoder_val) 'in'] )
        im=frame2im(getframe(gca));
        text('units','pixels','position',[100
50],'fontsize',24,'color','w','string','string',[ 'Depth: ' num2str(depth)
'in'] )
        im=frame2im(getframe(gca));
        image(im)
        imwrite(im,['GatorRail Defect' int2str(error_count)
'.jpg'],'jpg','Comment','Test Comment');
        error_count=error_count+1;
    end
    if(tline=='Ended')
          notdone=0;
    end
    pause(1)
  end
 end
fclose(serialport);
delete(serialport)
clear serialport
```

c. Main Code

```
/**
 * GatorRail Main Program File
 *
 * By Eitan Sisso
 */

#include <asf.h>
```

```
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ADC.h"
#include "sonar.h"
#include "sensor.h"
#include "behavior.h"
#include "PololuWheelEncoders.h"
#include "qdec_driver.h"

#define DbLedOn()          (PORTR.OUTCLR = 0x02)              //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()         (PORTR.OUTSET = 0x02)              //Turns the debug led off.
The led is connected with inverted logic
#define DbLedToggle()      (PORTR.OUTTGL = 0x02)              //Toggles the debug led
off.  The led is connected with inverted logic

#define CLOCK_DIV_bm  TC_CLKSEL_DIV64_gc               //Definitions for the Encoder
Timing
#define CLOCK_DIV     64

bool onoff=true;                                                      //Initializes
onoff which is toggled by the End of Rail Code
uint16_t lineCount = 2048;                                            //Number of
lines in the quadrature encoder.
int rev=-2;
long int encodervalue;
char teststring;
long double distance;
double conversionfactor=0.00125;
/*! Initializes Global frequency variable. */
uint16_t captureFreq  = 0;
uint16_t calcFreq     = 0;
uint16_t calcRPM      = 0;


int main (void)
{
       board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                                    motorInit() is declared within because by default you
the user should define what your
                                    motor setup is to prevent hurting the Epiphany.  You
can do this by
                                  */
       RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
       DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
       uartInit(&USARTC0,115200);//as can be seen in the schematic.  This uart is
connected to the USB port.  This function initializes this uart
```

```
	uartInit(&USARTE1,9600);//as can be seen in the schematic.  This uart is connected
to the Xbee port.  This function initializes this uart
	ADCsInits();//this function initializes the ADCs inside the Xmega
	sei();
	LCDInit();
	//This function initializes encoders
	//and end program/gives error if there are any problems
	if(!QDEC_Total_Setup(&PORTD,0,false,0,EVSYS_CHMUX_PORTD_PIN0_gc,false,EVSYS_QDIRM_
00_gc,&TCC0,TC_EVSEL_CH0_gc,lineCount)){
		fprintf(&lcd_str,"\r GatorRail has \n Encoder Error");
		return 0;
	}
	QDEC_TC_Freq_Setup(&TCD0, TC_EVSEL_CH2_gc, EVSYS_CHMUX_PORTD_PIN0_gc,
CLOCK_DIV_bm);

	PMIC.CTRL |= PMIC_LOLVLEN_bm;

	// CALIBRATION ROUTINE
	fprintf(&lcd_str,"\r GatorRail is \n    Calibrating");
	if(!sensorCalibrate()){
		fprintf(&lcd_str,"\r Calibration Error \n Check Sensors"); // if sensors
are not within expected values this returns an error
		_delay_ms(2000);
		for (int i = 0; i < 100; i++)
		{
		sensorRead();
		fprintf(&lcd_str,"\r Lft Mdl Rght\n %d %d
%d",leftdial,middledial,rightdial);
		_delay_ms(400);
		}
		return 0;
	}
	fprintf(&lcd_str,"\r Lft Mddl Rght\n %d %d %d",calibratedleft, calibratedmiddle,
calibratedright);
	_delay_ms(1000);
	fprintf(&lcd_str,"\r GatorRail is \n  READY TO GO");
	uint16_t i,j;
	_delay_ms(500);
	drive();

	while (onoff)
	{
		fprintf(&lcd_str,"\r GatorRail is \n Scanning");
		sensorRead();
		//fprintf(&lcd_str,"\r Lft Mdl Rght\n %d %d
%d",leftdial,middledial,rightdial);  //Displays Dial Indicator Sensor Data
		if ((TCD0.INTFLAGS & TC0_CCAIF_bm) !=  0) {
			encodervalue=(TCC0_CNT);
		} //Updates the encoder value
		/*
		fprintf(&lcd_str,"\r %d \n %d",encodervalue,rev);
		fprintf(&Xbee_str,"\r %d mm",encodervalue);
		*/
		if(DefectFinder())
		{
			Found(distance,depth);
		}
		onoff= EndOfRail();
		_delay_ms(50);
	}


	return 0;
}

ISR(TCC0_OVF_vect){	//This is the interrupt for when encoder overflows (counts revs)
	rev=rev+1;
}
```

d. Behaviors Code

```
/*
 * behavior.c
 *
 * Created: 4/8/2012 8:14PM
 *  Author: Eitan
 */

#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"
#include "sensor.h"
#include "behavior.h"
#include "qdec_driver.h"

void drive(void){
	setMotorDuty(1,800,MOTOR_DIR_FORWARD_gc);
	setMotorDuty(2,800,MOTOR_DIR_FORWARD_gc);
}

void stop(void){
	setMotorDuty(1,1024,MOTOR_DIR_BRAKE_gc);
	setMotorDuty(2,1024,MOTOR_DIR_BRAKE_gc);
	_delay_ms(1000);
}
bool DefectFinder(void){
	if ((leftdial<(calibratedleft-350)) || (rightdial<(calibratedright-350)) ||
(middledial<(calibratedmiddle-350)))
		{
			stop();
			//assign depth value
			//Left Sensor Depth (in.) = (decimal value of converter voltage -
873.06) / 10629
```

```
                    //Middle Sensor Depth (in.) = (decimal value of converter voltage -
783.07) / 10150
                    //Right Sensor Depth (in.) = (decimal value of converter voltage -
617.67) / 10252

                    if (leftdial<(calibratedleft-350)) depth=(calibratedleft-leftdial-
873.06)/10629;
                    else if (rightdial<(calibratedright-350)) depth=(calibratedright-
rightdial-617.67)/10252;
                    else depth=(calibratedmiddle-middledial-783.07)/10150;

                    return true;
              }
       else return false;
}
void Found(double distance,double depth){
              fprintf(&lcd_str,"\r FOUND!");
              drive();
              _delay_ms(375); //Gets robot in position for picture
              stop();
              sendreport(distance,depth);
              setMotorDuty(2,1024,MOTOR_DIR_FORWARD_gc); //Turns Brights On for Picture
              _delay_ms(2000);
              setMotorDuty(2,0,MOTOR_DIR_FORWARD_gc);//Turns Brights Off
              //Backs the robot up
              setMotorDuty(1,1024,MOTOR_DIR_BACKWARD_gc);
              setMotorDuty(2,1024,MOTOR_DIR_FORWARD_gc);
              _delay_ms(100);
              stop();
              drive();
}
bool EndOfRail(void){
       if (irfront< calibratedirfront*.75)
              {
                    stop();
                    fprintf(&lcd_str,"\r Rail Analysis \n Complete");
                    fprintf(&Xbee_str,"Ended, %d, %d",0,0);
                    return false;
              }
       else return true;
}
void sendreport(double distance,double depth){
       fprintf(&Xbee_str,"Found, %d, %d",distance,depth);
}
```